IT507 Advanced Image Processing

Assignment 2

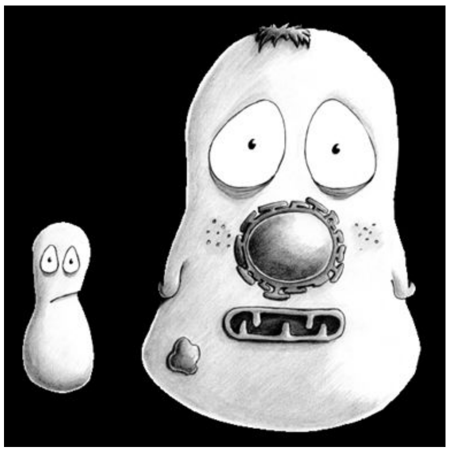
Darshil Patel(202011034)
Vaidik Patel(202011038)
Mayank Patel(202011045)

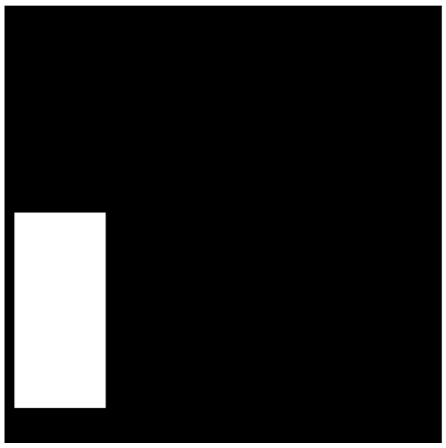
1. Consider Fig.1 and remove the larger object from the image. [Hint: Create a mask and apply arithmetic operation].

Answer:

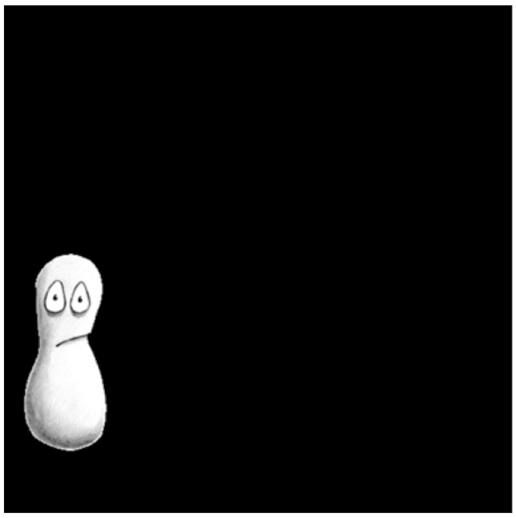
- First of all we create a mask containing white pixels (value 1) in the position of the smaller object else are black pixels (value 0).
- Then we perform element wise multiplication between image and mask. it gives us the image's pixel value where mask's pixel value is 1(white) else it gives 0 (black).
- The mask contains white pixels only at the position of the smaller object, so the multiplication process removes the larger object making its pixel black.

Image and mask:





Result:

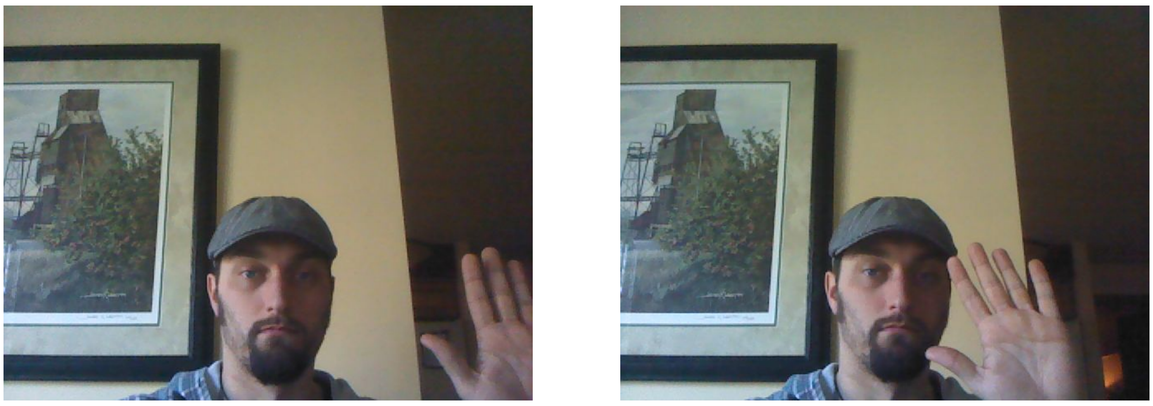


2. Consider Fig.2, and find out whether a motion has occurred between (a) and (b). [Hint: Find out the percentage of change in pixels.]

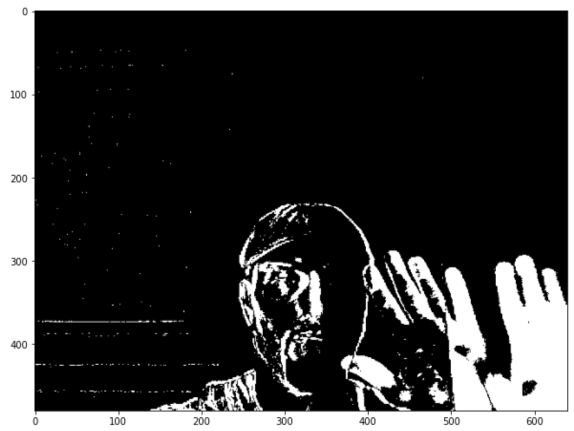
Answer:

- To check if any motion has occured in the images, we have to check if a change has occurred in the image. To do so we simply subtract the two images. This will give us a difference image. If the pixel value in the difference image is less than some threshold then we understand that nothing has changed but if the pixel value is greater than or less then threshold then some change has occurred. We called this threshold value as a change tolerance threshold value.
- we calculate the percentage of the changed pixels and if this percentage is greater than some threshold then we can conclude that a motion has occurred.
- If we mark change as 1 then we can clearly see where the change is happening.

Input Images:



Output Image:



Result:

- for the change tolerance threshold value 20, 10.63% pixels changes which shows the motion is happening between two pictures.

3. Implement an algorithm to find out the number plate of Fig.3(a) within the image (Fig. 3(b)). [Hint: Use the image matching algorithm and find out the normalized cross-correlation for each pixel].

Answer:

- Here we are using the cv2 library for matching templates.
- The main algorithm is based on for every pixel it finds a coefficient pixel value by using **normalized cross correlation** formula.
- In code we are using inbuilt function **cv2.matchTemplate()**. In that function we have three arguments, first is the real image where we want to match the template, second is the template image and third one is which matching method we use. In our case we are using **cv2.TM_CCORR_NORMED** option for normalized cross correlation.
- For every pixel function returns coefficient value which we can calculate by formula of normalized cross correlation. Below result showing the **cv2.matchTemplate()** method's result.

Normalized cross correlation result:

-
  ```
  [[0.8502626  0.8505249  0.85079443 ... 0.7830146  0.7830391  0.7837871 ]
  [0.84991175 0.85040134 0.8509349  ... 0.78125036 0.781092   0.7816344 ]
  [0.84945714 0.8499943  0.8507294  ... 0.77983314 0.7792694  0.7794166 ]
   ...
  [0.71857595 0.7168521  0.71509755 ... 0.6192716  0.62061405 0.6220063 ]
   [0.7198459  0.7181365  0.7163905  ... 0.6182316  0.6196285  0.6210132 ]
  [0.7209953  0.71930504 0.71756566 ... 0.6172824  0.6186943  0.6200755 ]]
  ```

- Now if we want to check which pixels value match with the template image then we first find the maximum value from these above values. For that in code we are using threshold value set **0.99**.

Result:

- (array([301], dtype=int64), array([305], dtype=int64))
- Above result is location of matching pixel. And from using that we draw rectangular. And Finally, we get the image below.